МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №1

Дисциплина: «Конфигурационное распределенное управление ПО»

Тема: «Основы работы с Docker»

Вариант 5

Выполнила: студентка 3 курса,

группы ИВТ-б-о-21-1

Диченко Дина Алексеевна

Ставрополь 2023

Цель занятия: Научить студентов использовать основные команды Docker для управления контейнерами и понимать их назначение.

Практическая часть:

Задание 1. Основы Docker

- Загрузите образ Ubuntu с Docker Hub.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
aece8493d397: Pull complete
Digest: sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fecac8176c6e496e5c1f05f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

- Создайте и запустите контейнер на основе этого образа.

- Войдите в созданный контейнер и выполните команду ls , чтобы просмотреть файлы внутри контейнера.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker run -it ubuntu
root@5da999af7cb8:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@5da999af7cb8:/#
```

Задание 2. Управление контейнерами и образами

- Загрузите образ Nginx с Docker Hub.

```
uper@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
 docker pull nginx:latest
atest: Pulling from library/nginx
f7ce2fa46ab: Pull complete
b16c94bb686: Pull complete
a59d19f9c5b: Pull complete
ea27b074f71: Pull complete
6edf33e2524: Pull complete
4b1ff10387b: Pull complete
17357831967: Pull complete
igest: sha256:4d2f2056993a84b7b9832c7612c5900c26ecd908bd2baacc8c18f6abbc8e0770
tatus: Downloaded newer image for nginx:latest
ocker.io/library/nginx:latest

uper@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
```

- Создайте контейнер на основе этого образа и пробросьте порт 8080 контейнера на порт 80 хоста.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker run -p 8080:80 -d nginx
786d3d5768b779e4ea3308ba281304ed79e4af496be440291920c121e557bc75
```

- Посмотрите список активных контейнеров и убедитесь, что ваш контейнер работает.

```
uper@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
 docker ps
ONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                  NAMES
15458efe3ba        nginx               "/docker-entrypoint.…"   5 minutes ago       Up 5 minutes        0.0.0.0:8080->80/tcp   musing_cohen
```

- Остановите и удалите контейнер.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker stop musing_cohen
musing_cohen

super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker rm musing_cohen
musing_cohen
```

Задание 3. Мониторинг и управление контейнерами

- Запустите контейнер с именем "my_container".

```
uper@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
 docker run --name my_container -d nginx
5ce8dd42b366c8505001ee2ebe5d5fdb335ee1e72278a2e46ff3c3d31b21b2d2
```

- Используя команду docker ps , убедитесь, что контейнер запущен.

```
uper@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
 docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
              NAMES
5ce8dd42b366        nginx               "/docker-entrypoint.…"   8 seconds ago       Up 6 seconds        80/tc
p             my_container
```

- Остановите контейнер.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker stop my_container
my_container
```

- Проверьте его статус снова и убедитесь, что он остановлен.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
          NAMES
```

- Удалите контейнер.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker rm my_container
my_container
```

Задание 4. Удаление образов и оптимизация дискового пространства

- Загрузите образы Ubuntu и Alpine с Docker Hub.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
96526aa774ef: Pull complete
Digest: sha256:eece025e432126ce23f223450a0326fbebde39cdf496a85d8c016293fc851978
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

- Создайте контейнеры на основе обоих образов.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name c_alpine -d alpine
f8e25d7e7cc79266bf3ab88ceeb05ba7981def36d797b049d95a34e131d9a2c5

super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name c_ubuntu -d ubuntu
460079fe0345a47abf9b2347a79e01ee7f88ae84ae8a73afb7300171e3843382
```

- Убедитесь, что контейнеры запущены и работают.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
460079fe0345        ubuntu              "/bin/bash"         29 seconds ago      Exited (0) 26 seconds ago                       c_ubuntu
f8e25d7e7cc7        alpine              "/bin/sh"           47 seconds ago      Exited (0) 43 seconds ago                       c_alpine
```

- Удалите образ Ubuntu.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker rmi -f ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fecac8176c6e496e
5c1f05f
Deleted: sha256:e4c58958181a5925816faa528ce959e487632f4cfd192f8132f71b32df2744b4
```

- Проверьте, что образ Ubuntu больше не существует, но образ Alpine остался на системе.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
460079fe0345        e4c58958181a        "/bin/bash"         3 minutes ago       Exited (0) 2 minutes ago                        c_ubunt
f8e25d7e7cc7        alpine              "/bin/sh"           4 minutes ago       Exited (0) 2 minutes ago                        c_alpin
```

Задание 5. Взаимодействие с контейнером

- Запустите контейнер с именем "my_container" в фоновом режиме.

```
super@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name my_container -d nginx
95f5029e2b5c6f29923125ed37cbac2172144161137f2afe7b76aaf2967d032b
```

- Используя команду docker exec , выполните команду ls -l /app внутри контейнера.

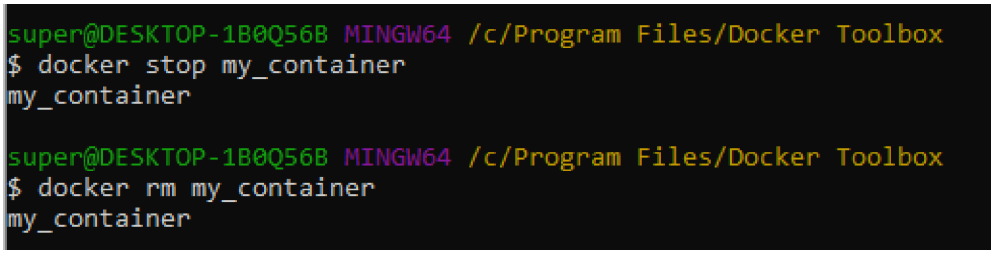
```
uper@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
 docker exec my_container ls -l /app
s: cannot access '/app': No such file or directory

uper@DESKTOP-1B0Q56B MINGW64 /c/Program Files/Docker Toolbox
 docker exec my_container ls -l
otal 60
rwxrwxrwx   1 root root    7 Nov 20 00:00 bin -> usr/bin
rwxr-xr-x   2 root root 4096 Sep 29 20:04 boot
rwxr-xr-x   5 root root  340 Nov 22 02:08 dev
rwxr-xr-x   1 root root 4096 Nov 21 09:05 docker-entrypoint.d
rwxrwxr-x   1 root root 1620 Nov 21 09:05 docker-entrypoint.sh
rwxr-xr-x   1 root root 4096 Nov 22 02:08 etc
rwxr-xr-x   2 root root 4096 Sep 29 20:04 home
rwxrwxrwx   1 root root    7 Nov 20 00:00 lib -> usr/lib
rwxrwxrwx   1 root root    9 Nov 20 00:00 lib32 -> usr/lib32
rwxrwxrwx   1 root root    9 Nov 20 00:00 lib64 -> usr/lib64
rwxrwxrwx   1 root root   10 Nov 20 00:00 libx32 -> usr/libx32
rwxr-xr-x   2 root root 4096 Nov 20 00:00 media
rwxr-xr-x   2 root root 4096 Nov 20 00:00 mnt
rwxr-xr-x   2 root root 4096 Nov 20 00:00 opt
r-xr-xr-x 135 root root    0 Nov 22 02:08 proc
rwx------   2 root root 4096 Nov 20 00:00 root
rwxr-xr-x   1 root root 4096 Nov 22 02:08 run
rwxrwxrwx   1 root root    8 Nov 20 00:00 sbin -> usr/sbin
rwxr-xr-x   2 root root 4096 Nov 20 00:00 srv
r-xr-xr-x  13 root root    0 Nov 22 02:08 sys
```

- Выполните команду ps aux внутри контейнера, чтобы увидеть список запущенных процессов.

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.3  0.0   4624  3612 pts/0    Ss   19:19   0:00 /bin/bash
root         9  0.0  0.0   7060  1668 pts/0    R+   19:20   0:00 ps aux
```

- Остановите и удалите контейнер.



Контрольные вопросы:

1. Что делает команда docker pull?

Используется для загрузки образа контейнера с Docker Hub или другого репозитория. Это важный этап при работе с контейнерами, так как он позволяет получить образ, который затем можно использовать для создания и запуска контейнеров.

2. Какой синтаксис используется для загрузки образа с Docker Hub с помощью docker pull?

docker pull <имя_образа>:<тег>

3. Как можно просмотреть список всех доступных образов на системе с помощью docker images?

docker images

4. Какой ключ используется для просмотра образов в формате таблицы с docker images?

docker images --format "table {{.Repository}}\t{{.Tag}}\t{{.Size}}"

5. Как создать и запустить контейнер с использованием docker run?

docker run [опции] <имя_образа> [команда] [аргументы]

6. Как пробросить порт при запуске контейнера с docker run ?

docker run -p 8080:80 nginx

7. Как изменить имя контейнера при его создании с помощью docker run?

docker run --name my_container -d nginx

8. Как создать контейнер в фоновом режиме с docker run?

Docker run -d nginx

9. Какая команда используется для просмотра активных контейнеров на системе?

docker ps

10. Какие опции могут использоваться с docker ps для отображения остановленных контейнеров?

Docker ps -a

11. Как можно просмотреть список всех контейнеров, включая остановленные, с docker ps?

docker ps -a

12. Что делает команда docker start?

Команда docker start в Docker используется для запуска остановленных контейнеров. Она позволяет вам возобновить выполнение контейнера, который был ранее приостановлен или остановлен.

13. Какой синтаксис используется для запуска остановленного контейнера с docker start?

docker start my_container

14. Как запустить контейнер в фоновом режиме с docker start?

docker start my_container

15. Что делает команда docker stop?

Используется для остановки работающего контейнера. Это важная команда, которая позволяет контролировать жизненный цикл контейнеров.

16. Как остановить контейнер по его имени с помощью docker stop?

docker stop my_container

17. Как принудительно остановить контейнер с docker stop ?

Docker stop -f my_container

18. Что делает команда docker rm ?

Используется для удаления контейнера, который был остановлен.

19. Как удалить контейнер по его ID с использованием docker rm?

docker rm 1234567890

20. Как удалить несколько контейнеров сразу с docker rm ?

docker rm container1 container2

21. Что делает команда docker rmi ?

Используется для удаления образов контейнеров с вашей системы. Это важная команда, которая позволяет освобождать дисковое пространство и управлять образами на вашей системе.

22. Как удалить Docker-образ по его имени и тегу с помощью docker rmi ?

docker rmi ubuntu:20.04

23. Как удалить несколько Docker-образов сразу с docker rmi ?

docker rmi image1 image2

24. Как выполнить команду внутри работающего контейнера с docker exec?

docker exec

25. Как выполнить команду внутри контейнера в интерактивном режиме с docker exec?

docker exec -it my_container /bin/bash

26. Как выполнить команду с использованием определенного пользователя внутри контейнера с docker exec?

docker exec -u 1000 my_container whoami

27. Какой ключ используется для запуска команды в фоновом режиме с docker exec?

docker exec -d my_container my_command

28. Как выполнить команду внутри контейнера с именем вместо ID с docker exec?

docker exec -it $(docker ps -q -f "name=my_container") /bin/bash

29. Как передать аргументы при выполнении команды с docker exec?

Docker exec -it $(docker ps -q -f “name=my_container”) /bin/bash

30. Как проверить список доступных команд и опций для docker exec?

docker exec [опции] <имя_или_ID_контейнера> <команда> [аргументы]

31. Как передать переменную окружения в контейнер при его запуске?

docker exec --help

32. Какой ключ используется для запуска контейнера в фоновом режиме с командой docker run?

docker run -e MYSQL_ROOT_PASSWORD=my-secret-pw mysql

33. Как проверить статус выполнения контейнеров на системе с помощью docker ps?

docker run -d nginx

34. Как завершить выполнение контейнера без его удаления?

docker ps -s

35. Каким образом можно удалить все остановленные контейнеры с системы?

docker stop my_container

36. Что делает опция -a при использовании docker ps?

docker rm $(docker ps -aq)

37. Что означает опция -q при выполнении docker ps?

Добавление опции -a позволяет просматривать все контейнеры, включая те, которые были остановлены.

38. Как принудительно удалить контейнер с флагом -f?

Добавление опции –q выводит только ID контейнеров.

39. Какой Docker-образ и какую команду можно использовать для создания контейнера с базой данных PostgreSQ?

docker rm -f my_container

40. Какой ключ используется для выполнения команды внутри контейнера в интерактивном режиме?

docker run –name postgres_container postgres

41. Какой ключ используется для выполнения команды внутри контейнера в интерактивном режиме?

docker exec -it my_container

42. Какой ключ можно использовать для передачи ID пользователя при выполнении команды внутри контейнера?

С опцией -u мы указываем ID пользователя, от имени которого будет выполнена команда.

Вывод: в результате выполнения работы были изучены основные команды Docker для управления контейнерами.